CMPSC 465 Data Structures & Algorithms
Spring 2024 Mehrdad Mahdavi and David Koslicki

HW 4

Due February 9, 10:00 pm

Instructions: You are encouraged to solve the problem sets on your own, or in groups of up to five people, but you must write your solutions strictly by yourself. You must explicitly acknowledge in your write-up all your collaborators, as well as any books, papers, web pages, etc. you got ideas from.

Formatting: Each problem should begin on a new page. Each page should be clearly labeled with the problem number. The pages of your homework submissions must be in order. You risk receiving no credit for it if you do not adhere to these guidelines.

Late homework will not be accepted. Please, do not ask for extensions since we will provide solutions shortly after the due date. Remember that we will drop your lowest two scores.

This homework is due Friday, February 9, 10:00 pm electronically. You need to submit it via Gradescope (Class code XX7RVV). Please ask on Campuswire about any details concerning Gradescope and formatting.

1. (20 pts.) **Same Negative Value as Index.** Given a sorted zero-indexed array A of n (possibly negative) distinct integers, you want to find out whether there is an index i for which $A[i] = -i$. Give an algorithm for this problem with $O(\log n)$ running time.

(a) Describe your algorithm in words and explain why it's correct.

(b) Analyze the running time of your algorithm.

2. (20 pts.) **Matrix Squaring vs Matrix Multiplication.** The square of matrix A is its product with itself, AA.

(a) Show that five multiplications are sufficient to compute the square of a 2×2 matrix.

(b) What is wrong with the statement below on an algorithm for computing the square of an $n \times n$ matrix?
"Use a divide-and-conquer approach as in Strassen's algorithm, except that instead of getting 7 subproblems of size $n/2$, we now get 5 subproblems of size $n/2$ thanks to part (a). Using the same analysis as in Strassen's algorithm, we can conclude that the algorithm runs in $O(n^{\log_2 5})$ time."

(c) In fact, squaring matrices is no easier than multiplying them. Show that if $n \times n$ matrices can be squared in $O(n^c \log n)$ time for a constant c, then any $n \times n$ matrices can be multiplied in $O(n^c \log n)$ time.
(*Hint:* First, show that squaring a $2n \times 2n$ matrix can also be done in $O(n^c \log n)$ time. Then, think about how to obtain the result of multiplying two $n \times n$ matrices from the result of squaring a $2n \times 2n$ matrix.)

3. (20 pts.) **Majority Element.** An array $A[0,\ldots,n-1]$ is said to have a majority element if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as image files, say.) However you can answer questions of the form: "is $A[i] = A[j]$?" in constant $O(1)$ time.

(a) Show that if x is a majority element in the array then x is a majority element in the first half of the array or in the second half of the array.

(b) Show how to check in $O(n)$ time if a candidate element x is indeed a majority element.

(c) Put these observation together to design a divide-and-conquer algorithm whose running time is $O(n \log n)$. (Hint: Use the master theorem to analyze its running time.)

(d) We will now design a linear-time algorithm. Here's another divide-and-conquer approach:

- Pair up the elements of A arbitrarily, to get $n/2$ pairs. (If n is odd, we keep the singleton unpaired element but only to break a potential tie; that is, this element can't be used to overrule a majority.)
- Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them.

Let L be the elements left after this procedure. Show that L has at most $\lceil n/2 \rceil$ elements, and that if A has a majority element, then L has the same majority element.

(e) Use parts (d) and (b) to design an algorithm for this problem with $O(n)$ running time.

4. (20 pts.) **Recurrence relations.** Solve the following recurrence relations and give the tightest correct upper bound for each of them in O notation. Assume that $T(1) = O(1)$.

(a) $T(n) = 27T(n/3) + 17n^3$

(b) $T(n) = 2T(n/7) + \sqrt{n}$

(c) $T(n) = T(n-1) + c^n$, where $c > 1$ is a constant

(d) $T(n) = T(n-1) + n^c$, where $c \geq 1$ is a constant

(e) $T(n) = 7T(n/4) + n$

(f) $T(n) = T(n/2) + 2.25^n$

(g) $T(n) = 49T(n/36) + n^{3/2} \log n$

(h) $T(n) = T(\frac{n}{3}) + T(\frac{n}{6}) + T(\frac{n}{12}) + n$ (*Hint: Assume $T(k) \leq ck$ for some constant $c > 0$ for all $k < n$. Prove that $T(n) \leq c_1 n$ for some constant $c_1 > 0$ by induction.*)

(i) $T(n) = T(3n/7) + T(4n/7) + \Theta(n)$. (*Hint: Assume $T(k) \leq ck \log k$ for some constant $c > 0$ for all $k < n$. Prove that $T(n) \leq c_1 n \log n$ for some constant $c_1 > 0$ by induction.*)

(j) $T(n) = \sqrt{n}T(\sqrt{n}) + 11n$. (*Hint: carefully unfold the recurrence $k = \log\log n$ steps and argue that the resulting terms are all $O(n \log\log n)$.*)

5. (20 pts.) **Modified Hadamard matrices.** The matrices $H_0, H_1, H_2, \ldots$ are defined as follows:

- H_0 is the 1×1 matrix $[1]$
- For $k > 0$, H_k is the $2^k \times 2^k$ matrix

$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ -H_{k-1} & -H_{k-1} \end{bmatrix}$$

So, for example, $H_1 = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$ and

$$H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Show that if v is a column vector of length $n = 2^k$, then the matrix-vector product $H_k v$ can be calculated using $O(n \log n)$ operations. Assume that all the numbers involved are small enough so that basic arithmetic operations like addition and multiplication take unit time. Explain how your solution has running time $O(n \log n)$.

6. (0 pts.) **Acknowledgments.** The assignment will receive a 0 if this question is not answered.
(a) If you worked in a group, list the members of the group. Otherwise, write "I did not work in a group."
(b) If you received significant ideas about the HW solutions from anyone not in your group, list their names here. Otherwise, write "I did not consult with anyone other than my group members."
(c) List any resources besides the course material that you consulted in order to solve the material. If you did not consult anything, write "I did not consult any non-class materials."